STEALTHSWAP : PRIVACY PRESERVING ETHEREUM TRANSACTIONS

Elliot.S
stealthy-ai@protonmail.com

Yi Qun
yiqun-chu@protonmail.com

ABSTRACT

StealthSwap is a new on-chain protocol that allows users to send and receive shielded payments through stealth addresses and is easily integrated into the existing ecosystem through its SDK. StealthSwap is implemented as a set of oracle smart contracts that broadcast and publish encrypted notes on-chain. The encrypted notes allow participants to use stealth addresses to make shielded payments that are untraceable to external observers. The StealthSwap protocol utilizes the OWL Token as a fee mechanism to prevent privacy attacks and congestion of the oracle contracts. OWL Token works as an incentive mechanism for token holders to participate in the long-term security for the StealthSwap protocol, maintain the anonymity and reliability assumptions, and provide rewards for staking OWL.

1 Introduction

1.1 The Ethereum Protocol and Smart Contracts

Ethereum is an open blockchain platform that lets anyone build and use decentralized applications via blockchain technology. As is the case with Bitcoin, no one controls or owns the Ethereum protocol – it is an open-source project built by many people from around the world. Unlike the Bitcoin protocol, Ethereum was designed to be adaptable and flexible. It is easy to create new applications on the Ethereum platform. Smart contracts are decentralized tamper-proof applications that live on the blockchain. Their execution and function are triggered by message calls (transactions) and, as applications, smart contracts can use these external calls to execute other smart contracts. Smart contracts can operate as oracles by publishing events, but also as algorithmic entities able to receive and send tokens without any third-party interaction. These properties are what makes smart contracts an attractive alternative to centralized solutions, especially since the blockchain serves as a tamper-proof platform for insuring and verifying the publication of data.

1.2 Privacy on The Ethereum Network

A primary example of privacy is anonymity, or the privacy of identity. In the context of public blockchains, anonymity refers to the ability for parties to exchange something (i.e. money, tokens, or data) without the need to reveal identity-related information about themselves or their previous transactions and their counter-parties. While this is only a basic facet of privacy, it has become important as blockchain technology has evolved. Cryptocurrencies like Bitcoin and Ethereum are currently able to have their blockchains parsed, also known as mined, across all transactions that have existed on their respective blockchain. If this information is coupled with information from a third party, such as a fiat-to-crypto on ramp is discovered for a wallet address, or if information is unintentionally revealed due to a wallet owner's lapse in judgement, the identity of the wallet holder, and potentially their counter-parties, can easily be derived.

The need for privacy : As blockchain technology continues to rapidly emerge into today's everyday society across every field imaginable, protection of consumer and enterprise data is a critical function in the preservation of sensitive data. While there are many cases where public blockchains provide users and viewers with a ledger of all historical transactions and their participants may be ideal, the protection of privacy for consumers and enterprises has become evidently necessary, especially in purely financial transactions. All users, whether they be enterprises or consumers,

require privacy in the form of transaction data. For example, the goods bought or received, for how many and how much, addresses of the payee and the payer, or personal financial information, to name a few.

The traceability issues of decentralized ledgers : The lack of privacy on the existing Ethereum network presents a critical issue due to the fact that accounts are far easier to trace than UTXOs. For example, if you buy a cup of coffee using your Ethereum wallet, the barista doesn't need to know how much money you have left in it or who your previous transactions were with.

2 Flaws with Existing Ethereum Anonymity Solutions

2.1 Existing Obfuscation Methods

Mixers are services for making funds anonymous for users based on scrambling transactions together, effectively obfuscating the origin source of a specific blockchain based token. Unfortunately, mixers can only function with multiple transactions happening at the same time by utilizing a set of fake transactions, called "Mix-ins," plus several real transactions, which effectively results in wallets possessing tokens where the origin cannot be determined.

As an example When someone sends a transaction for an amount such as 5 ETH, one must wait for several other people to also send the same amount of Ethereum in order to mix and obfuscate the source of these coins. Mixers will then send several transactions to a multitude of addresses, which ultimately sum to the original units inputted being redistributed to each user, respectively. This method of obfuscation is ultimately dependent upon many users wishing to make the same request simultaneously. It is not totally fool proof and the original funds trying to be mixed can still be linked back to the original address via chain analysis.

2.2 Barriers Towards Anonymity

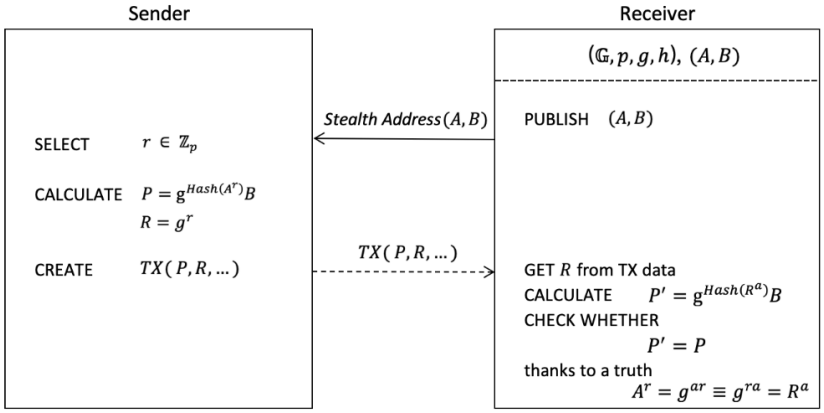
The Ethereum ledger tracks user's balances using the *accounts* abstraction. In comparison with *UTXOs*, accounts work like a balance sheet and they're long-lived in the *state trie*.This has the effect that transactions are more easily tracked to a specific holder as demonstrated in [7]. On the other hand, UTXOs are represented as coins, which are harder to pin-point, but their origin and history is still known to all external observers. Due to transactions in the UTXO abstract, the concepts of inputs and outputs are easily amenable to *mixing*. Accounts, on the other hand, require a far more difficult process to mix. Thus comes the need for a different approach, one that does not suffer from the short-comings and restrictions of mixers.

The EVM Costs The current state-of-the-art mixer on the Ethereum platform is [9]. To mix a certain amount users send a *hash commitment* and funds to a smart-contract. This *commitment* is added to a growing list of deposits that represent the anonymity set at that step. When users wish to withdraw they have submit a *proof* that they have knowledge of the *pre-commitment*. This constitutes the *zkSNARK* part of the proof. The cost of verifying SNARK proofs on-chain is high. According to [8], gas prices fluctuate between 900k for deposits and 700k for withdrawals. For reference, gas costs for pre-compile *ECC* and *Pairing* operations, see [10]

The risks of cutting-edge cryptography Using cutting-edge cryptography has always come with high-risks. While zkSNARKs offer a singular low-level primitive to build *zero-knowledge authentication* protocols, it also lends itself to the risk of exploits due to the complexity of pin-pointing the exact assumptions of the prover-verifier aspects.

3 The StealthSwap Protocol

The StealthSwap protocol builds on well-tested cryptography primitives. Stealth addresses, Pedersen Commitments [2], and the Elliptic Curve Diffie–Hellman (ECDH). A way to think of a stealth address is a one-time safety deposit box. Only the receiver, the holder of the private key, can open the box to see what is inside of it and claim it. A stealth address differs from traditionally utilized addresses as they are generated randomly for one-time usage. The ECDH is a cryptographic protocol used to generate a shared secret between users, allowing bi-directional encrypted communication.A stealth address enables the sender and receiver to non-interactively make payments with almost perfect anonymity. Its implementation and utilization are presented here:



3.1 Preliminaries

An Elliptic Curve E over a prime field $\mathbf{F_p}$ is defined by the Weierstrass equation :

$$E(\mathbf{F_p}) : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{1}$$

where a_1, a_2, a_3, a_4 are field elements and the curve discriminant $\Delta \neq 0$, the set of solutions $(x, y) \in E(\mathbf{F_p})$ defines the group of points on the elliptic curve. We define scalar multiplication for points by :

$$kG = \underbrace{G + G \cdots + G}_{\text{k-times}} \tag{2}$$

The scalar multiplication is the fundamental operation in elliptic curve based cryptographic protocols such as the Elliptic Curve Diffie-Hellman (ECDH) key agreement. The parameters used in the remaining of this document is for the curve **secp256k1** as defined in [11]. For more details about elliptic curve cryptography, the interested reader is referred to [1].

3.2 Stealth Addresses

The Concept of a Stealth Address Stealth Addresses have been present since 2011 but only incorporated in privacy based ledgers in 2014; for a history see [3]. The main purpose of stealth addresses is untraceable transfers by *breaking the link* from the sender and receiver.

One-Time Use Stealth Addresses The simplest form of a stealth address is built using ECDH. The sender samples a random scalar *r* and computes a one time public key using the base point of the protocol's curve and the receiver's public key *P*. Given $r \in \mathbf{Z_p}$ and a *secure hash function* Hash, the receiver Alice publishes a public key A from its keypair (A, a).

$$P = \text{Hash}(r * A) * G \tag{3}$$

The sender, Bob, transfers funds to the new public key P and sends Alice $R = r * G$. On her end, Alice will scan the transactions until she can find a receiving public key she can *unlock* using the private key P' computed as follows:

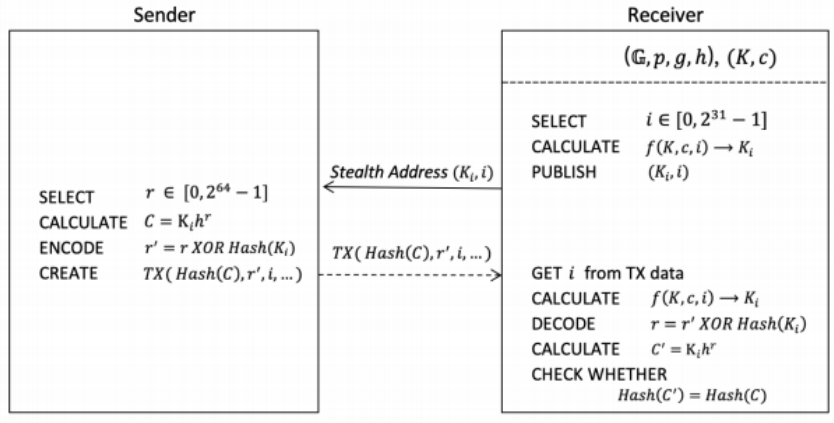
$$P' = \text{Hash}(a * R) * G \tag{4}$$

The cons of this scheme is that *r* is known to both the sender and receiver, and the complexity of scanning all transactions in the blockchain renders it impractical for users.

Improvements The scheme could be further improved as was done by [3]. In the new scheme *Fig-2* we use instead a hierarchical deterministic address scheme and Pedersen commitments as the destination key. The advantages of this improved version is the compact data and the minimal additional payload. The cons are the requirements of a secured communication channel to share the address information. To solve this issue we introduce a secure channel information with data guarantees.

3.3 Oracle Smart Contracts

A solution to secure channel in the second scheme is to introduce an oracle *Master* smart contract. The role of the contract is to broadcast an *encrypted note* that signals a shielded payment event. The *Master* oracle contract thus works



as an informer that a payment has been done to a *stealth address*. The oracle publishes an *encrypted note* that can only be decrypted by the receiver's private key. This process effectively solves the issue of tainting user's keys and the existence of a *secure channel* without any third party requirements. It also lowers the complexity of scanning the blockchain to just the emitted events.

Encrypted Note for ERC-20 Transfers An encrypted note announcing a new token transfer is a tuple of encrypted data published on-chain that effectively announces a transfer of ERC-20 tokens to the stealth address.

$$Note(Address, Amount, Token, InitVector, CipherText, PublicKey, MAC) \tag{5}$$

Encrypted Note for ETH Transfers For Ethereum transactions the note is a similar tuple without the ERC-20 contract's address as Ether is the default currency of the network.

$$Note(Address, Amount, InitVector, CipherText, PublicKey, MAC) \tag{6}$$

ENS Domains In order to execute the protocol in a non-interactive model, the protocol uses ENS names to fetch *public keys* of users that are used to initiate and execute the protocol. On a first usage, users publish a signed message to their ENS name in the form of a text record as described in [5]. During each transfer or receive operation, the user's public key is used to *encrypt* the *encrypted note* published on-chain, then the user's private key is used to *decrypt* the note. Instead of requiring interaction between users, senders make a payment to an *ENS Name*, the protocol implementation takes care of generating, encrypting and broadcasting the payment *note* to the smart contract and the newly generated *stealth address* becomes the holder of the funds. To receive the payment to a secondary address, users only need their *ENS Name* to authenticate themselves. This process happens non-interactively and off-chain, effectively reducing all on-chain metadata that could taint their privacy, and simplifies user experience.

Advantages and On-Chain Costs Contrary to mixers, *shielded payments* using the StealthSwap protocol require no waiting times, no deposits, and no custody of funds. The protocol functions as a set of middle men and all received funds can be sent to other addresses immediately. While users can keep the fund in their *stealth address*, this renders them vulnerable to malicious attackers that track all oracle broadcasts.

To solve this users can simply *withdraw* funds from their stealth address to another address they control. This operation can be done instantly once the transaction has confirmed on-chain. We recommend users that this should be considered as part of privacy hygiene.

The protocol is also homogeneous towards Ethereum transactions. In other words, it can be used to transfer ERC-20 tokens, withdraw funds from Uniswap, and interact or integrate with any active Ethereum based application.

This is the first privacy preserving scheme of its kind on Ethereum that bridges the need of instantaneous transactions and privacy preservation.

The current privacy guarantees of the protocol is that all transactions are private to external observers, and no information leakage is possible except for the two interacting parties.

Fees The cost of a *shielded payment* is exactly that of a regular Ethereum transaction plus a *fee* paid to the *Oracle* contract in *OWL Tokens*. The fee paid to the *Oracle* is a protective measure that prevents congestion and malicious

attacks against the protocol. The usage of token fees is to build a long-term incentive mechanism for user adoption, and participation in the protocol's sustainability and long-term availability.

Security Without this fee, threat actors can congest the oracle contract trough dummy payments rendering it unusable and possibly tainting stealth addresses trough dust attacks.

3.4 Further Improvements

Pedersen Commitments Depending on protocol usage, an upgrade towards a more advanced stealth address scheme might be possible. While the current protocol uses minimal data, it can be further truncated by publishing *commitments* instead of whole notes.

zkSNARKs With the current integration of zkSNARK friendly op-codes in [12], the protocol can be further improved to incorporate *zkSNARK* proofs instead of *encrypted notes*. This improvement will further hide the *stealth address* from the sender, reducing the knowledge to the receiver only. The implementation of such scheme currently can be done off-chain. With the next release of the protocol, it will be introduced as the main way to make *shielded payments* in favor of the current protocol. The current obstacle faced is the high price for verifying *zkSNARK* proofs on-chain.

4 The OWL Token and mechanism design for privacy protocols

The OWL Token is StealthSwap's own *protocol security* token. It functions similarly to the mining reward in *Bitcoin*, rewarding the oracle contract through transaction fees. The collected transaction fees of the oracle are distributed as *staking rewards* to token holders. This serves as both an incentive mechanism for user participation and liquidity provision to power the protocol, guaranteeing user anonymity and the security of the on-chain oracle contracts.

Gas Station Relayers Since stealth addresses are unfunded, withdrawing funds from a one-time address to another requires users to pay for the Gas fees currently demanded by the Ethereum network. In order to avoid tarnishing the anonymity of any address and preserving the broken link, users will utilize the funds held to pay for the transaction fees. In the case of ERC-20 tokens, swapping them for ETH and using the Gas Station Network as described in [6].

Protection Against Malicious Actors In order to prevent threats and malicious actors from spamming the protocol with faulty transactions that intend to slow down the broadcasting of encrypted notes, a transaction fee is paid to the contract itself in OWL tokens.

Collected Fees Distribution The collected fees are then distributed to the staking pool, which rewards OWL holders who stake their tokens, which provides liquidity for the protocol users to transact through. Additionally, 40 percent of all OWL Tokens supply will be awarded to the staking pool linearly across the first four years of staking in order to supplement the incentives to provide liquidity to the network.

4.1 Integration with the Ecosystem

The protocol architecture is composed of *smart contracts* that act as on-chain encrypted data oracles and a software integration kit.

StealthPay Using the SDK built, *StealthPay* will be the first dApp to leverage the StealthSwap protocol for seamless shielded payments. The SDK integrates all the ease of use and security into a simple UI.

Integration Using the SDK, any dApp can integrate shielded payments for its users. This move was made with the effective purpose of making privacy preserving protocols simpler to access for users. We strongly believe that privacy is a requirement, a need for all Ethereum users.

Challenges There are two major challenges with privacy preserving protocols in blockchains. One of the challenges is statistical analysis and the second is the integration with other protocols and developer experience. With StealthSwap, we aimed towards solving both of these challenges with the least overhead for users. Given the recent explosive growth of DeFi protocols, which still need a major privacy upheaval, our goals were to optimize user privacy and composability.

5 Final Words

In this paper, we have presented the StealthSwap protocol and the backing infrastructure used to deliver privacy preserving payments to Ethereum users. As usage of the blockchain rises, users will naturally need solutions to make their transactions private. StealthSwap offers a highly efficient, user friendly, privacy preserving solution to solve the existing issue.

References

[1] Darrel Hankerson, Alfred J. Menezes, Scott Vanstone "Guide to Elliptic Curve Cryptography". Springer-Verlag, New York. 2005.8

[2] Torben Pryds Pedersen Non-interactive and information-theoretic secure verifiable secret sharing

[3] Yu, G. (2020, July 3). Blockchain Stealth Address Schemes. Retrieved from ttps://eprint.iacr.org/2020/548.pdf

[4] Inoue, M., Lau, J., Eigenmann, D., and Johnson, N. (n.d.). ENS: Decentralized Naming for Wallets, Addresses, and More. Retrieved September, from https://ens.domains/

[5] Richard Moore "EIP-634: Storage of text records in ENS [DRAFT]", Ethereum Improvement Proposals, no. 634, May 2017. [Online serial]. Available: https://eips.ethereum.org/EIPS/eip-634

[6] Gas Network Station https://docs.opengsn.org/learn/

[7] Béres, F., Seres, I. A., Benczúr, A. A., and Quintyne-Collins, M. (2020). Blockchain is Watching You: Profiling and Deanonymizing Ethereum Users. arXiv preprint arXiv:2005.14051.

[8] Dune Analytics. Analysis of Ethereum Data.

[9] Tornado.cash. A zkSNARK Mixer for Ethereum Transactions. https://tornado.cash

[10] Gavin Wood The Ethereum Yellowpaper https://ethereum.github.io/yellowpaper/paper.pdf

[11] Standards for Efficient Cryptography Group, SEC2: Recommended Elliptic Curve Domain Parameters version 2.0, http://www.secg.org/sec2-v2.pdf

[12] Connor, Konda, Westland EIP-1922: zk-SNARK Verifier Standard https://eips.ethereum.org/EIPS/eip-1922